三阶Crank-Nicolson格式即其与JuLiLi论文的结合

三阶时间差分格式:

考虑等距分布的时间差分的函数值:u^{n+1},u^{n},u^{n-1},假设其在$t^{n-1+\alpha}$处存在Taylor展开式:

其中$\alpha\in(1,2)$表示时间区间$[t^n,t^{n+1}]$上的某一个点;

$$
\begin{aligned}
u^{n+1}&=u^{n-1+\alpha}+(2-\alpha)\tau\frac{\partial u}{\partial t}\left(t^{n-1+\alpha}\right)+\frac{(2-\alpha)^2}{2}\tau^2\frac{\partial^2 u}{\partial t^2}\left(t^{n-1+\alpha}\right)+\frac{(2-\alpha)^3}{6}\tau^3\frac{\partial^3 u}{\partial t^3}\left(t^{n-1+\alpha}\right)+o\left(\tau^4\right);\\
u^{n}&=u^{n-1+\alpha}+(1-\alpha)\tau\frac{\partial u}{\partial t}\left(t^{n-1+\alpha}\right)+\frac{(1-\alpha)^2}{2}\tau^2\frac{\partial^2 u}{\partial t^2}\left(t^{n-1+\alpha}\right)+\frac{(1-\alpha)^3}{6}\tau^3\frac{\partial^3 u}{\partial t^3}\left(t^{n-1+\alpha}\right)+o\left(\tau^4\right);\\
u^{n-1}&=u^{n-1+\alpha}+(0-\alpha)\tau\frac{\partial u}{\partial t}\left(t^{n-1+\alpha}\right)+\frac{(0-\alpha)^2}{2}\tau^2\frac{\partial^2 u}{\partial t^2}\left(t^{n-1+\alpha}\right)+\frac{(0-\alpha)^3}{6}\tau^3\frac{\partial^3 u}{\partial t^3}\left(t^{n-1+\alpha}\right)+o\left(\tau^4\right);
\end{aligned}
$$

那么假设存在u^{n+1},u^{n},u^{n-1}的一个线性表出,使得(1)中形成一个:

$$
span\left\{u^{n-1},u^{n},u^{n+1}\right\}=\lambda(\alpha)\tau\frac{\partial u}{\partial t}\left(t^{n-1+\alpha}\right)+o\left(\tau^4\right)\tag{2}
$$

的状态;

等价于下列齐次线性方程组存在非平凡根:

$$
\begin{bmatrix}1&0&0\\0&\frac{\tau^2}{2}&0\\0&0&\frac{\tau^3}{6}\end{bmatrix}\begin{bmatrix}1&1&1\\(2-\alpha)^2&(1-\alpha)^2&(0-\alpha)^2\\(2-\alpha)^3&(1-\alpha)^3&(0-\alpha)^3\end{bmatrix}\begin{bmatrix}\chi\\\beta\\\gamma\end{bmatrix}=\mathbf{0}\tag{3}
$$

因此(3)的核空间完全取决于α所在的矩阵的行列式,因此计算得到其有两个根:

$$
\alpha=1\pm\frac{\sqrt{3}}{3}\tag{4}
$$

这两个根都有用,因为对于一个初值问题,我们只知道u^0,而不知道u^1,u^2,那么按照离散化的方程(组)在这两点成立,联立方程解出这两个值才能保证对应的三阶精度.且在这两点都搞定以后,后面每一步只需要求解较大根了.

根据我们的假设,选取较大的根,并且我们断言:

$$
(2-\alpha)\chi+(1-\alpha)\beta+(0-\alpha)\gamma\neq0\tag{5}
$$

如若不然,则下面的方程成立:

$$
\begin{bmatrix}1&1&1\\(2-\alpha)&(1-\alpha)&(0-\alpha)\\(2-\alpha)^2&(1-\alpha)^2&(0-\alpha)^2\end{bmatrix}\begin{bmatrix}\chi\\\beta\\\gamma\end{bmatrix}=\mathbf{0}\tag{6}
$$

即一个范特蒙德矩阵存在非平凡齐次解,那么必须满足其中的:$2-\alpha,1-\alpha,-\alpha$存在相等的情况,显然这是不行的.

因此我们得到了一个三阶数值格式:

$$\frac{\chi u^{n+1}+\beta u^{n}+\gamma u^{n-1}}{\left[(2-\alpha)\chi+(1-\alpha)\beta+(0-\alpha)\gamma\right]\tau}=\frac{\partial u}{\partial t}\left(t^{n-1+\alpha}\right)+o\left(\tau^{3}\right) \tag{7}$$

那么对于空间差分时,只要获得某点的函数值的近似而不需要考虑关于时间导数的近似,因此容易多了,也即找出上述的u^{n+1},u^{n},u^{n-1}的一组线性标出能够使得时间一阶导数和二阶导数系数为0即可:

$$\begin{bmatrix}1&0&0\\0&\tau&0\\0&0&\frac{\tau^{2}}{2}\end{bmatrix}\begin{bmatrix}1&1&1\\(2-\alpha)&(1-\alpha)&(0-\alpha)\\(2-\alpha)^{2}&(1-\alpha)^{2}&(0-\alpha)^{2}\end{bmatrix}\begin{bmatrix}\chi\\\beta\\\gamma\end{bmatrix}=\begin{bmatrix}\neq 0\\0\\0\end{bmatrix} \tag{8}$$

注意到这里求解时首先把(8)看出两个方程:

$$\begin{bmatrix}1&0&0\\0&\tau&0\\0&0&\frac{\tau^{2}}{2}\end{bmatrix}\xi=\begin{bmatrix}\neq 0\\0\\0\end{bmatrix};\tag{9}$$

$$\begin{bmatrix}1&1&1\\(2-\alpha)&(1-\alpha)&(0-\alpha)\\(2-\alpha)^{2}&(1-\alpha)^{2}&(0-\alpha)^{2}\end{bmatrix}\begin{bmatrix}\chi\\\beta\\\gamma\end{bmatrix}=\xi;$$

也即得出ξ是和τ无关的量,且第一个元素要求是非零元;

那么假设这个非零元就是1,那么由于第二个方程的系数矩阵是范特蒙德矩阵,因此解存在唯一.

为了不引起混淆,我们记(9)的唯一解为(λ,μ,η)

因此就以热方程为例,下面的BDF-2格式是时间三阶方法:

$$\frac{\chi u^{n+1}+\beta u^{n}+\gamma u^{n-1}}{\left[(2-\alpha)\chi+(1-\alpha)\beta+(0-\alpha)\gamma\right]\tau}+\Delta_{h}\left(\lambda u^{n+1}+\mu u^{n}+\eta u^{n-1}\right)=0 \tag{10}$$

(10)中:

$$\begin{bmatrix}\chi\\\beta\\\gamma\end{bmatrix}=\begin{bmatrix}1\\-4\frac{\alpha-1}{2\alpha-1}\\\frac{2\alpha-3}{2\alpha-1}\end{bmatrix};\begin{bmatrix}\lambda\\\mu\\\eta\end{bmatrix}=\begin{bmatrix}\frac{\alpha^{2}-\alpha}{2}\\-\alpha^{2}+2\alpha\\\frac{\alpha^{2}}{2}-\frac{3\alpha}{2}+1\end{bmatrix};\tag{11}$$

一个很简单的验证

为了验证这个方法确实是三阶精度的时间差分格式,首先考虑使用常微分方程来判断其收敛阶:

对于常微分方程的Cauchy问题:

$$\frac{dx}{dt}=1-x^{2},x\left(0\right)=0\Leftrightarrow x\left(t\right)=\tanh(t) \tag{12}$$

第一步通过四阶龙格库塔方法得到:假设求解区域是$[0,1]$,步长是τ,那么第一步:

$$\begin{aligned}&K_{1}=1;K_{2}=f\left(0+\tfrac{\tau}{2}\right)=1-\tfrac{\tau^{2}}{4};\\&K_{3}=f\left(0+\tfrac{\tau}{2}\left(1-\tfrac{\tau^{2}}{4}\right)\right)=1-\left(\tfrac{\tau}{2}-\tfrac{\tau^{3}}{8}\right)^{2};K_{4}=1-\left(\tau\left[1-\left(\tfrac{\tau}{2}-\tfrac{\tau^{3}}{8}\right)^{2}\right]\right)^{2};\\&u_{1}=\tfrac{\tau}{6}(K_{1}+2K_{2}+2K_{3}+K_{4});\end{aligned} \tag{13}$$

后面的每一步都能够用(10)的方法计算一个一元二次方程.

同时需要和Vanilla的BDF-2进行对比:给出python程序:

```
import numpy as np
alpha = 1 + np.sqrt(3)/3
chi = 1
beta = -4*(alpha-1)/(2*alpha-1)
gamma = (2*alpha-3)/(2*alpha-1)
lamb = (alpha**2 - alpha)/2
mu = -alpha**2 + 2*alpha
eta = 1/2*alpha**2 - 3*alpha/2 + 1
deno = (2-alpha)*chi + (1-alpha)*beta + (0-alpha)*gamma
def give_initial_state(tau):
    K1 = 1;
    K2 = 1 - 1/4*tau**2
    K3 = 1 - (tau/2*K2)**2
    K4 = 1 - (tau*K3)**2
    return tau/6*(K1 + 2*K2 + 2*K3 + K4)
def bdf_2_3ord(un,un_1,tau):
    coef_un_pls = chi/deno
    coef_lfs = (beta*un + gamma*un_1)/deno
    a = tau*(lamb**2)
    b = 2*lamb*tau*(mu*un+eta*un_1) + coef_un_pls
    c = coef_lfs - tau + tau*((mu*un+eta*un_1)**2)
    delta = b**2 - 4*a*c
    if delta < 0:
        print("illegal inputs")
        return -1
    else:
        c1 = (-b + np.sqrt(delta))/2/a
        c2 = (-b - np.sqrt(delta))/2/a
        if np.abs(c1-un) < np.abs(c2-un):
            return c1
        else:
            return c2
def bdf_2_vanilla(un,un_1,tau):
    a = 2*tau
    b = 3
    c = -4*un + un_1 - 2*tau
    delta = b**2 - 4*a*c
    if delta < 0:
        print("illegal inputs")
        return -1
    else:
        c1 = (-b + np.sqrt(delta))/2/a
        c2 = (-b - np.sqrt(delta))/2/a
        if np.abs(c1-un) < np.abs(c2-un):
            return c1
        else:
            return c2


if __name__ == "__main__":
```

```
#   具有三阶精度的bdf-2:
#   粗网格
    N = 20
    tau = 1/N
    res = np.zeros((N+1))
    x = np.linspace(0,1,N+1)
    res[1] = give_initial_state(tau)
    for k in range(2,N+1):
        res[k] = bdf_2_3ord(res[k-1],res[k-2],tau)
    re1_max = np.max(np.abs(res - np.tanh(x))**2)
    re1_mse = np.mean(np.abs(res - np.tanh(x))**2)
#   细网格
    N = 40
    tau = 1/N
    res = np.zeros((N+1))
    x = np.linspace(0,1,N+1)
    res[1] = give_initial_state(tau)
    for k in range(2,N+1):
        res[k] = bdf_2_3ord(res[k-1],res[k-2],tau)
    re2_max = np.max(np.abs(res - np.tanh(x))**2)
    re2_mse = np.mean(np.abs(res - np.tanh(x))**2)
    print(f"具有三阶级精度的bdf-2格式加细一倍后的极大误差减小为原来的:1/{re1_max/re2_max}")
    print(f"具有三阶级精度的bdf-2格式加细一倍后的平方误差减小为原来的:1/{re1_mse/re2_mse}")
    print(re1_max,re1_mse,re2_max,re2_mse)
#    消融实验
#   Vanilla的bdf-2:
#   粗网格
    N = 20
    tau = 1/N
    res = np.zeros((N+1))
    x = np.linspace(0,1,N+1)
    res[1] = give_initial_state(tau)
    for k in range(2,N+1):
        res[k] = bdf_2_vanilla(res[k-1],res[k-2],tau)
    re1_max = np.max(np.abs(res - np.tanh(x))**2)
    re1_mse = np.mean(np.abs(res - np.tanh(x))**2)
#   细网格
    N = 40
    tau = 1/N
    res = np.zeros((N+1))
    x = np.linspace(0,1,N+1)
    res[1] = give_initial_state(tau)
    for k in range(2,N+1):
        res[k] = bdf_2_vanilla(res[k-1],res[k-2],tau)
    re2_max = np.max(np.abs(res - np.tanh(x))**2)
    re2_mse = np.mean(np.abs(res - np.tanh(x))**2)
    print(f"341格式的bdf-2格式加细一倍后的极大误差减小为原来的:1/{re1_max/re2_max}")
    print(f"341格式的bdf-2格式加细一倍后的平方误差减小为原来的:1/{re1_mse/re2_mse}")
    print(re1_max,re1_mse,re2_max,re2_mse)
```

具有三阶级精度的bdf-2格式加细一倍后的极大误差减小为原来的:1/59.268732124944755

具有三阶级精度的bdf-2格式加细一倍后的平方误差减小为原来的:1/58.70609726213347

7.2543289500157565e-12 1.3720450042201928e-12 1.2239723526939743e-13 2.3371422530333786e-14

341格式的bdf-2格式加细一倍后的极大误差减小为原来的:1/14.0235989048689

341格式的bdf-2格式加细一倍后的平方误差减小为原来的:1/13.41510626255866

1.991703388582696e-07 1.0296686718899066e-07 1.4202512508334718e-08 7.675441787320722e-09

发现在这个简单问题上,Vannila BDF-2大概有3.8阶精度,而具有三阶精度的bdf-2有5.7阶精度

偏微分方程的验证:

给定一维一个热传导方程,具有显示表达式,那么考虑三阶格式的bdf-2和Vanilla bdf-2的对比:

$$\frac{\partial u}{\partial t}-\frac{\partial^2 u}{\partial x^2}=0, x\in(-1,1), t>0 \tag{14}$$

其在初边值为:$u(x,0)=\sin(\pi x), u(-1,t)=u(1,t)=0$;时具有解析解:$u=\exp\left(-\pi^2 t\right)\sin\pi x$

那么在考虑vanilla bdf-2格式时,每一步离散化后成为:

$$\frac{3u^{n+1}-4u^n+u^{n-1}}{2\Delta t}=\frac{u_{j+1}^{n+1}-2u_j^{n+1}+u_{j-1}^{n+1}}{h^2}; \tag{15}$$

因此对应采用快速求解器求解.

Matlab编程:

```
N = 128;
>> x = (-1:2/N:1).';
>> tau = 0.01;
>> r = tau/((2/128).^2);
a = 3 + 4*r;
>> b = -2*r;
A = sparse([1:N-1,2:N-1,1:N-2],[1:N-1,1:N-2,2:N-1],[ones(1,N-1)*a,ones(1,2*N-4)*b],N-1,N-1);
y0 = sin(pi*x);
LP = sparse([1:N+1,2:N+1,1:N],[1:N+1,1:N,2:N+1],[ones(1,N+1)*(-2),ones(1,2*N)],N+1,N+1);
LP(1,1) = 0;LP(1,2) = 0;LP(N+1,N) = 0;LP(N+1,N+1) = 0;
h = 2/N;
K1 = LP*y0/(h^2);
K2 = LP*(y0 + tau*K1/2)/(h^2);
K3 = LP*(y0 + tau*K2/2)/(h^2);
K4 = LP*(y0 + tau*K3)/(h^2);
u1 = y0 + (K1 + 2*K2 + 2*K3 + K4)*tau/6;
plot(x,u1)
```

一次算两步的修正:

热方程的数值结果说明,网比必须小于1才能稳定迭代,因此把(4)中两个根都利用起来:形成下面的隐格式:

$$\frac{\chi_i u^{n+2}+\beta_i u^{n+1}+\gamma_i u^n}{\left[(2-\alpha_i)\chi_i+(1-\alpha_i)\beta_i+(0-\alpha_i)\gamma_i\right]\tau}=\Delta_h\left(\lambda_i u^{n+2}+\mu_i u^{n+1}+\eta_i u^n\right), i=1,2; \tag{16}$$

为了方便表示,下面把分母和网比都写成特殊形式:

$$d_i = \left[\left(2-\alpha_i\right)\chi_i + \left(1-\alpha_i\right)\beta_i + \left(0-\alpha_i\right)\gamma_i\right]; r = \frac{\tau}{h^2}; \tag{17}$$

$$\begin{bmatrix} \frac{\chi_1}{d_1} & \frac{\beta_1}{d_1} \\ \frac{\chi_2}{d_2} & \frac{\beta_2}{d_2} \end{bmatrix} \begin{bmatrix} u^{n+2} \\ u^{n+1} \end{bmatrix} = r \begin{bmatrix} \lambda_1 & \mu_1 \\ \lambda_2 & \mu_2 \end{bmatrix} \begin{bmatrix} Lu^{n+2} \\ Lu^{n+1} \end{bmatrix} + r \begin{bmatrix} \eta_1 \\ \eta_2 \end{bmatrix} Lu^n - \begin{bmatrix} \frac{\gamma_1}{d_1} \\ \frac{\gamma_2}{d_2} \end{bmatrix} u^n \tag{18}$$

因此能够求解(18)的关键是求解广义特征值问题:具体方法描述为:为寻找满足:

$$\left(u^{n+2} + \omega u^{n+1}\right) = L\left(u^{n+2} + \omega u^{n+1}\right) \tag{19}$$

的实数ω:考虑:

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} := \begin{bmatrix} \frac{\chi_1}{d_1} & \frac{\beta_1}{d_1} \\ \frac{\chi_2}{d_2} & \frac{\beta_2}{d_2} \end{bmatrix} \setminus \begin{bmatrix} \lambda_1 & \mu_1 \\ \lambda_2 & \mu_2 \end{bmatrix} \tag{20}$$

那么:代入(19)得到:

$$u^{n+2} + \omega u^{n+1} = rL\left[\left(a_{11} + \omega a_{21}\right)u^{n+2} + \left(a_{12} + \omega a_{22}\right)u^{n+1}\right] \tag{21}$$

那么根据比例性质:

$$\frac{\omega}{1} = \frac{a_{12} + \omega a_{22}}{a_{11} + \omega a_{21}} \tag{22}$$

即一个一元二次方程,解得两个根:

通过数值验证出存在两个共轭复根.(因为(20)中的系数都是常数):

-2.0000 + 3.4641i -2.0000 - 3.4641i

那么(21)分别对于两个ω对应的线性方程组求解后得到的新的:

$$u^{n+2} + \omega u^{n+1} = res\left(1\right); u^{n+2} + \bar{\omega} u^{n+1} = res\left(2\right); \tag{23}$$

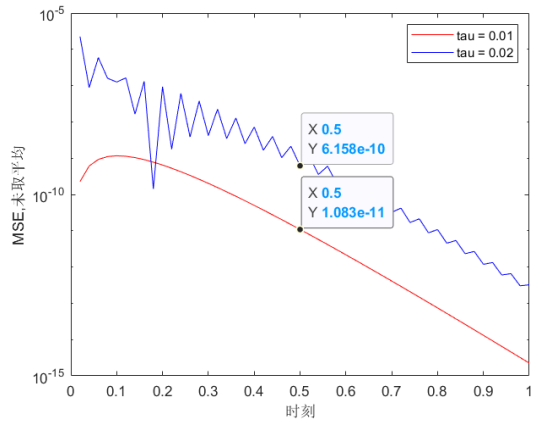
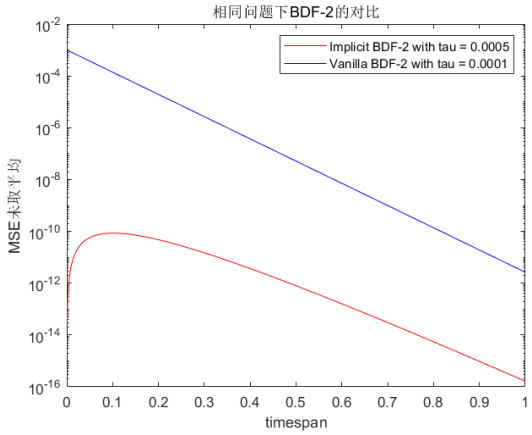
之后反解出u^{n+1}, u^{n+2}即可

或者写成kronecker积的形式:

$$\left[I - r\left(\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \otimes L\right)\right] \begin{bmatrix} u^{n+2} \\ u^{n+1} \end{bmatrix} = rhs; \tag{24}$$

这个格式是否稳定,首先数值验证:

$$rhs = r \begin{bmatrix} \frac{\chi_1}{d_1} & \frac{\beta_1}{d_1} \\ \frac{\chi_2}{d_2} & \frac{\beta_2}{d_2} \end{bmatrix} \setminus \begin{bmatrix} \eta_1 \\ \eta_2 \end{bmatrix} Lu^n - \begin{bmatrix} \frac{\chi_1}{d_1} & \frac{\beta_1}{d_1} \\ \frac{\chi_2}{d_2} & \frac{\beta_2}{d_2} \end{bmatrix} \setminus \begin{bmatrix} \frac{\gamma_1}{d_1} \\ \frac{\gamma_2}{d_2} \end{bmatrix} u_n \tag{25}$$

h = 2/2048时的热传导耗散问题的求解精度,(24)方法	(24)方法和3,-4,1格式的求解误差的对比.
	

(24)对应方程的快速求解器:

$$\begin{bmatrix} \frac{F^*}{n} & \\ & \frac{F^*}{n} \end{bmatrix} \begin{bmatrix} I - ra_{11}L & -ra_{12}L \\ -ra_{21}L & I - ra_{22}L \end{bmatrix} \begin{bmatrix} F & \\ & F \end{bmatrix} = \begin{bmatrix} \Lambda_1 & \Lambda_2 \\ \Lambda_3 & \Lambda_4 \end{bmatrix} \tag{26}$$

很显然四个对角阵组成的矩阵求逆的显式表达式是:

$$\begin{aligned} X_3 &= \left(\Lambda_2 - \Lambda_1(\Lambda_3)^{-1}\Lambda_4\right)^{-1}; \\ X_2 &= \left(\Lambda_3 - \Lambda_4(\Lambda_2)^{-1}\Lambda_1\right)^{-1}; \\ X_1 &= -(\Lambda_3)^{-1}\Lambda_4\left(\Lambda_2 - \Lambda_1(\Lambda_3)^{-1}\Lambda_4\right)^{-1}; \\ X_4 &= -(\Lambda_2)^{-1}\Lambda_1\left(\Lambda_3 - \Lambda_4(\Lambda_2)^{-1}\Lambda_1\right)^{-1}; \end{aligned} \tag{27}$$

并且(26)求逆后保持对角阵格式,因此可以仍然使用FFT.

(24)格式的稳定性分析:

由于这是个一推二的格式,因此必须去掉中间的u^{n+1}不考虑:

$$\left[\begin{pmatrix} I - a_{11}rL & -a_{12}rL \\ -a_{21}rL & I - a_{22}rL \end{pmatrix}\right] \begin{bmatrix} u^{n+2} \\ u^{n+1} \end{bmatrix} = \begin{bmatrix} r\tilde{\eta}_1 L u^n - \kappa_1 u^n \\ r\tilde{\eta}_2 L u^n - \kappa_2 u^n \end{bmatrix} \tag{28}$$

这里引入了新的符号,比对(25)可知其含义;

由于(28)中整体是可交换的,那么直接使用离散Fourier变换:

$$\sigma\left(u^{n+2}\right) = \frac{r^2s^2 + 3rs + 3}{r^2s^2 - 3rs + 3}, \sigma\left(u^{n+1}\right) = \frac{6 - r^2s^2}{2\left(r^2s^2 - 3rs + 3\right)}; \tag{29}$$

上述结果的得出通过matlab:

```
Alpha = [1+sqrt(3)/3,1-sqrt(3)/3];
chi = [1,1];
Beta = -4*(Alpha-1)./(2*Alpha-1);
Gamma = (2*Alpha-3)./(2*Alpha-1);
Lambda = (Alpha-1).*Alpha/2;
Mu = -Alpha.*(Alpha-2);
eta = (Alpha.^2)/2 - 3*Alpha/2 + 1;
d = ((2-Alpha).*chi + (1-Alpha).*Beta + (0-Alpha).*Gamma);
H = [chi(1)/d(1),Beta(1)/d(1);chi(2)/d(2),Beta(2)/d(2)];
A = H\[Lambda(1),Mu(1);Lambda(2),Mu(2)];
Eta = H\[eta(1);eta(2)];
Kappa = H\[Gamma(1)./d(1);Gamma(2)./d(2)];
syms s r
LP = [1 - A(1,1)*r*s,-A(1,2)*r*s;-A(2,1)*r*s,1-A(2,2)*r*s];
rhs = [r*Eta(1)*s-Kappa(1);r*Eta(2)*s-Kappa(2)].';
re = inv(LP)*rhs.';
simplify(re)


ans =
 (r^2*s^2 + 3*r*s + 3)/(r^2*s^2 - 3*r*s + 3)
    -(r^2*s^2 - 6)/(2*(r^2*s^2 - 3*r*s + 3))
```

这是不是和CN格式的稳定性分析很像?即$u^n->u^{n+2}$的推理是一个齐次多项式的比值,显然这个推理是稳定的,

因为Laplace算子的特征值s都是非正的.

JuLiLi论文中的思路:

我们上述讨论的都是线性方程,对于Allen-Cahn格式的非线性方程,保持稳定的话需要使用隐格式,且每一步求解时需要使用到Picard不动点迭代(牛顿迭代也行,但是显然不合适,计算Hessian阵显然不好算且消耗算力):

我们注意到,对于Allen-Cahn方程,Crank-Nicolson隐格式迭代法是下面的非线性方程组:

$$\begin{aligned}\frac{u^{n+1}-u^n}{\tau}&=\varepsilon^2\Delta_h\frac{u^{n+1}+u^n}{2}+\left[\frac{u^{n+1}-\left(u^{n+1}\right)^3}{2}+\frac{u^n-\left(u^n\right)^3}{2}\right]\\\frac{u^{n+1}-u^n}{\tau}&=\varepsilon^2\Delta_h\frac{u^{n+1}+u^n}{2}+\left[\frac{u^{n+1}-\left(u^{n+1}\right)^3}{2}+\frac{u^n-\left(u^n\right)^3}{2}\right]-\frac{1}{N^2}\sum_{i,j}^{N}\left(u_{i,j}-u_{i,j}^3\right)\end{aligned}\tag{30}$$

这里给的第二个是考虑守恒律形式的,但是本质方法都是把u^n固定,然后按照任意的一个u^{n+1}的时间显式二阶估计作为初始值,反复求解线性方程组,得到一个近似解,那么这个求解过程是线性变换(FFT)和非线性变换(三次算子与三次函数积分)交替进行的,恰好与神经网络的组合函数是吻合的。

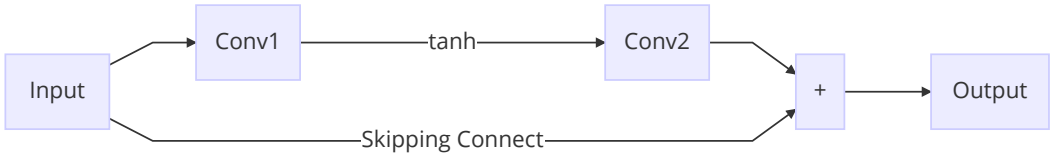
那么这个文章考虑构建了一个简单的cnn网络并且在输出层增加了有界限制器来防止输出数值太大.

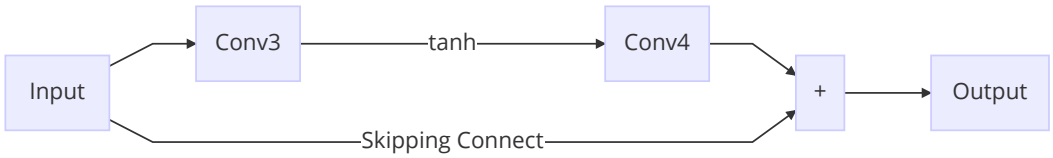
网络结构:

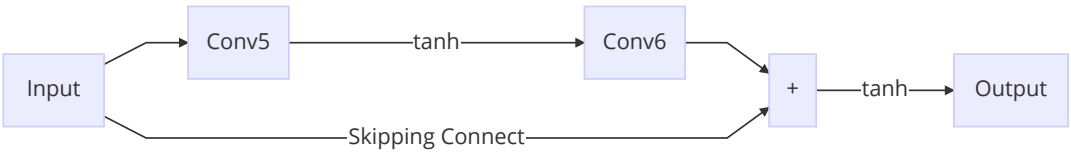
他所使用的网络结构是标准的总分总结构:

1.输入是首先使用卷积层conv1将输入的通道数从1增加到若干个,文件中名为:mid_planes

2.中间是三个ResBlock首尾相接,其中每个残差块定义为:



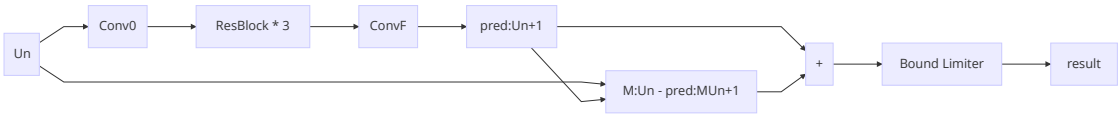




注意到这里使用到了ResNet中的跳跃连接,一种很常用的防止模型特征塌陷的手段.

3.最后使用到卷积层ConvF将mid_plane个通道归成一个通道并输出;

且保质量守恒的该模型多出了这个结构:消除多余的质量:



该网络的核心就是三个残差块,可调整的参数只有卷积通道数mid_planes.

损失函数:

文中的损失函数就是用于求解一般的Allen-Cahn方程的Crank-Nicolson格式,并且就采用MSE损失;

$$L_{ACNN-MSE}=\frac{1}{SN^2}\sum_{l=1}^{S}\sum_{i,j=1}^{N}\left[U_{n+1}^{i,j,(l)}-U_n^{i,j,(l)}-\frac{\Delta t}{2}\left(\varepsilon^2\Delta_h\left(U_{n+1}^{i,j,(l)}+U_n^{i,j,(l)}\right)+f\left(U_{n+1}^{i,j,(l)}\right)+f\left(U_n^{i,j,(l)}\right)\right)\right]^2 \quad (31)$$

这里没有写出质量守恒律格式的损失函数了,可以看原论文.并且这里多出的S指的是同时训练的数据集的数量:可以看成训练集总大小.因为数值算例中给出的初值是以符合均匀分布的乘积型区域上的随机数,这样可以提高模型的泛化能力,防止过拟合的发生.

训练策略:

文中给了1,2,3,4,5步骤,概括为:

1.将S分成p个batch,每个batch q个初始值;

2.按照顺序抽取batch,对于每个batch中的元素如下操作:

3.由u^0推理u^1,然后u^1推理u^2，直到时间步长达到a;文中在每个推理阶段训练b轮

4.处理完所有batch之后进入下一个epoch;

效果验证:

数值实验见原论文,但是其都是按照先训练前一部分时间,然后后序时间按照前后步的推理完成.

把三阶Crank-Nicolson思想嫁接到Julili文章中:

Julili文章的思想是好的,但是学习的目标仅仅只有二阶时间精度,我们都知道周期边界的空间精度可以使用快速Fourier变换达到任意有限阶,但时间精度受到时间差分格式的严格限制,那么如果我们考虑一推二形式训练是不是会更好呢?

先写出三阶Crank-Nicolson格式对于Allen-Cahn方程的离散化形式:

$$\begin{aligned}&\frac{\chi_i u^{n+2}+\beta_i u^{n+1}+\gamma_i u^n}{d_i\tau}=\varepsilon^2\Delta_h\left(\lambda_i u^{n+2}+\mu_i u^{n+1}+\eta_i u^n\right)+f\left(\lambda_i u^{n+2}+\mu_i u^{n+1}+\eta_i u^n\right);\\&i=1,2;f\left(u\right)=u-u^3;\end{aligned}\tag{32}$$

因此训练时就很粗暴地把损失函数设置为:

$$\frac{1}{2N^2}\sum_{i=1}^{2}\left\{\frac{\chi_i u^{n+2}+\beta_i u^{n+1}+\gamma_i u^n}{d_i}-\left[\tau\varepsilon^2\Delta_h\left(\lambda_i u^{n+2}+\mu_i u^{n+1}+\eta_i u^n\right)+\tau f\left(\lambda_i u^{n+2}+\mu_i u^{n+1}+\eta_i u^n\right)\right]\right\}^2\tag{33}$$

并且u^{n+2}是由u^n推理两次得到的结果.

我的方法相比BDF-3的优越性在于,减少了函数复合自身的次数,对于深度学习的训练能够节省显存的同时保证同样的精度,并且其中的u^{n+1}不作为输入,只作为输出,将原来的训练次数减半.